Fast Configuration of MEMS-Based Storage Devices for Streaming Applications

Mohammed G. Khatib and Hylke W. van Dijk
University of Twente
Enschede, the Netherlands
{m.g.khatib, h.w.vandijk}@utwente.nl

***Abstract*—An exciting class of storage devices is emerging: the class of Micro-Electro-Mechanical storage Systems (MEMS). Properties of MEMS-based storage devices include high density, small form factor, and low power. The use of this type of devices in mobile infotainment systems, such as video cameras is not at all obvious. We must explore their configuration and assess their benefit with respect to existing devices, such as Flash.**

In this paper, we study the configuration of the data layout of MEMS-based storage devices for predominately streaming applications. The configuration targets: the timing performance, energy consumption, and the capacity. We show that formatting the data layout based on knowledge of the expected streaming workload results in devices that consume at least 22% less energy than Flash and perform comparably.

We present a fast and effective configuration method to format the data layout. This method exploits the bimodal distribution of the request size in streaming applications. Using the fast method, we present a design rule for streaming environments: reducing the amount of prefetching of streaming data allows to reach configurations with small trade-offs between the design targets.

***Index Terms*—Secondary storage, Probe storage, Quality of service, Energy efficiency, Design space, Data layout.**

I. Introduction

An exciting class of storage devices is under development, called MEMS-based storage devices. These devices leverage Micro-Electro-Mechanical Systems (MEMS) [1], [2] for the actuation of a moving storage medium and for integration with electronic components. Properties of MEMS-based storage devices include high storage density, small form factor, and low power [3], [4]. Moreover, manufacturing is potentially economical, since MEMS-based storage devices can be made in existing, mostly written off, factories. In this paper, we address two issues of interest toward the integration of MEMS-based storage devices in the storage hierarchy of the computer system. The first issue is *configuring a MEMS-based storage device, so that it exhibits high performance and consumes little energy*. The second issue addresses *how the resulting configurations compete with alternative storage technologies*. We limit our treatment to mobile infotainment systems, such as handheld video cameras. These systems have a storage hierarchy, in which a main dynamic memory is directly connected to a non-volatile secondary-storage device. The storage device is in these systems optimized for streaming operations.

Mobile infotainment systems deploy Flash for their secondary storage. Today's Flash memories underwent several maturity stages with configurations that have been investigated for their integration in mobile infotainment systems. For MEMS-based storage such configurations are not all obvious. In this paper, we derive feasible configurations for the application of MEMS-based storage devices as secondary storage. We first explore the configuration space exhaustively, as a secondary result we demonstrate a fast method for their configuration that yields close to optimum results. Since MEMS-based storage devices are still in the development stage, we can extrapolate their feasibility for future systems.

In this work, we consider real-world traces from predominantly streaming applications. We have two contributions. (1) We show that it is possible to achieve data-layout configurations of MEMS-based storage devices that yield a comparable performance to Flash memories, yet consuming 22% less energy. Further, we show that MEMS-based storage devices demand a low amount of prefetching. Streaming real-world traces exhibit bimodality in their request-size distribution: a small request size, which is due to file system operations, and a large request size that is due to the streaming application. (2) We exploit this bimodality and show that the designer can effectively configure the data layout without the need for an exhaustive exploration of the design space. We present a configuration method based on the bimodality, recommending the designer to reduce the amount of prefetching to reach small trade-offs between the design targets of a MEMS-based storage device.

Although prototypes of MEMS-based storage exist already [3], MEMS-based storage devices are not publicly available yet. This inhibits us to experiment with a real MEMS-based storage device. Thus, the second best option is to use trace-driven simulation, which allows us to capture the activity of a real device in sufficient detail. To keep our study as realistic as possible we compare with empirical values from a flash card.

The main contribution of this work is the configuration method to format the data layout in streaming applications. Despite the fact that MEMS-based storage is still under development, we believe that our method will have a word later in real MEMS-based storage devices. The comparison figures we report on can be seen as an indication of the potential of this technology to compete with flash memory despite of its "mechanical" nature. Validation of these figures remains an open problem for future work to be tackled once a device materializes.

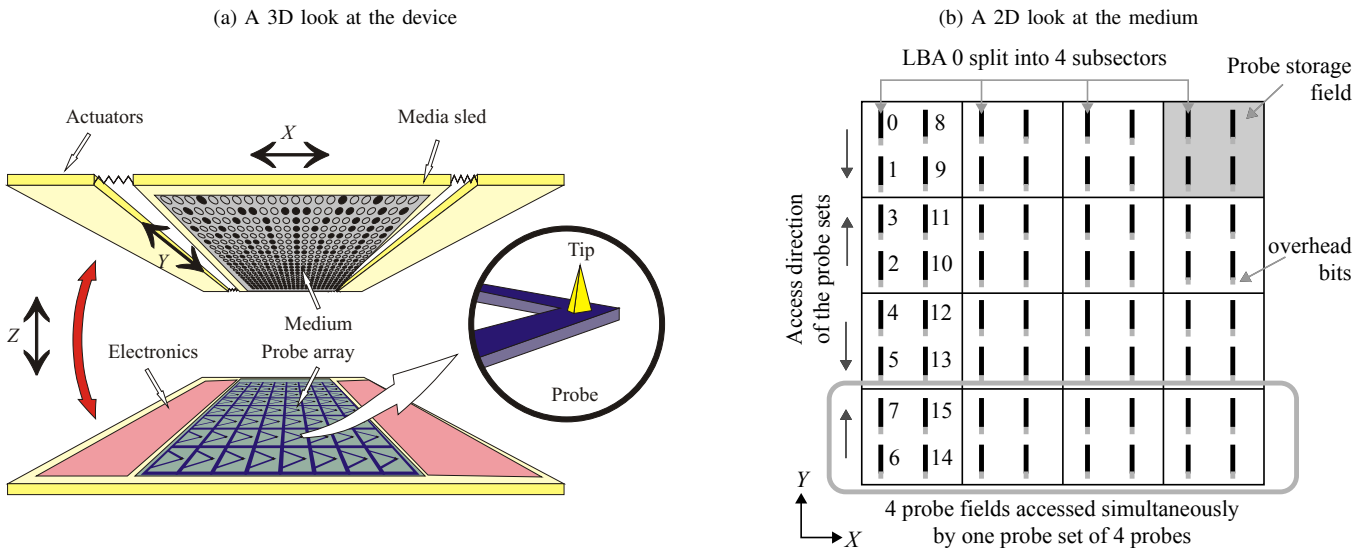


Fig. 1: Three- and two-dimensional views of a MEMS-based storage device. *(a) Two layers facing each other where the media sled is attached to springs that suspend it across the probe array. (b) The storage area of a simplified MEMS-based storage device consisting of* 4×4 *probe storage fields.*

In the following, Section II presents the design space, and introduces MEMS-based storage. Section III discusses the related work. Section IV introduces the configuration parameters of the data layout. In Section V, we detail our experimental methodology. Section VI compares MEMS-based storage to Flash memory. Section VII investigates the influence of prefetching on the quality of service of MEMS-based storage devices, and devises a fast and effective configuration method for streaming environments. Section VIII concludes.

II. Requirements and Basics

A handheld video camera is a concrete example of a mobile infotainment system. A camera must be affordable, must run for a reasonably long period of time, and must store a large amount of high-quality multimedia contents. Consequently, a stringent set of non-functional requirements are imposed on the storage device; Throughout this paper we focus on three design targets. The secondary storage device must:

1) consume little energy,
2) have a large storage capacity, and
3) exhibit short response times (for filesystem requests).

Note that the storage capacity represents the cost per bit. This is because retaining most of the device capacity at formatting, increases the effective capacity the user can exploit for a given paid price. The device must have low cost per bit. To compare, disk drives and Flash both offer large capacity solutions. Disk drives though cost \$0.50 per gigabit but also consume approximately 100 nJ/bit, generally too much energy for a mobile system, whereas Flash is energy efficient consuming approximately 10 nJ/bit, but costs about \$4 per gigabit.

A. MEMS-Based Storage

Several design models for MEMS-based storage have been proposed [1], [2], [5], [6]. Although these models adopt different storage and actuation techniques, they have a common architecture. A MEMS-based storage device consists of two distant physical layers, one above the other, as shown in Figure 1a. The top layer, called the *media sled*, is suspended by springs across the bottom layer, where the Z distance is maintained by nanopositioners. The bottom layer is a two-dimensional array of read/write probes or heads, called the *probe array*. For example, an IBM MEMS prototype [1] has a 64×64 probe array.

Bits can be recorded on a magnetic patterned medium as in μSPAM [6] and the CMU MEMStore [5]; a polymer medium as in the IBM MEMS device [1]; or a phase-change medium as in the Nanochip MEMS device [2]. The sled moves independently in the X, Y, and Z directions relative to the probe array. In all design models, each probe sweeps over a bounded area of the media sled, called the *probe (storage) field* as sketched in Figure 1b. Consequently, seek times shorten. Griffin *et al.* [7] explain the data layout in great detail.

The data layout assigns each sector a Logical Block Number (LBA) by which it is uniquely addressable. A relatively high (aggregate) data rate is attained by striping a sector across a probe set of several probes as shown in Figure 1b. Each probe accesses a subsector of a sector. Subsectors are augmented with additional bits to enable accessibility and buffering between successive sectors. To read data from or write to the medium, the media sled moves along the Y direction, along which data tracks lie as shown in Figure 1b. While accessing data, the X actuators keep the sled in position along the X direction on the accessed data track, counteracting the springs

restoring force. During inactivity, the springs hold the sled at its resting position, where every probe faces the centre position of its field.

III. Related Work

Several roles have been proposed for MEMS-based storage devices: as (1) a disk cache [8], (2) a streaming buffer and cache [9], and (3) a replacement for disk drives [10], and (4) a full and partial replacement of disks and the Non-Volatile Random Random Access Memory (NVRAM) in disk arrays [11].

Wang *et al.* [8] show that employing a MEMS-based storage device as a cache for disk drives with a capacity of just 3% of the disk capacity, the I/O subsystem exhibits the performance of MEMS-based storage devices 30% to 49% of the time. Rangaswami *et al.* [9] use MEMS-based storage devices as a buffer (and a cache) in streaming servers to reduce the demand for the expensive volatile DRAM. As a result, the DRAM buffering requirement decreases significantly up to 90% depending on the number of simultaneous streams. Schlosser *et al.* [10] replace disk drives by MEMS-based storage devices. They show that I/O stall times reduce by a factor of $4 - 74$ times over disks, and the overall application runtimes improve by a factor of $1.9 - 4.4$ times over disks. Uysal *et al.* [11] offer an extensive study of various roles of MEMS-based storage devices in disk arrays for server application. By varying the MEMS-to-disk cost ratio in the range $1 - 10$, they show that a full replacement of disks in a disk array improves the performance-to-cost ratio by a factor of $2 - 7$. A partial replacement results in a factor $2.5 - 5.0$, whereas a full replacement of the NVRAM has a factor of $2.1 - 4.2$.

To the best of our knowledge, this work is the first that investigates the deployment and configuration of MEMS-based storage devices as secondary storage in battery-powered streaming systems.

IV. Design parameters

The attainable data rate per probe in MEMS-based storage devices is limited by several factors including the resonance frequency [12] of the probe; the per-probe data rate is 40 Kbps in the present IBM prototype [1]. To elevate the performance, a MEMS-based storage device employs several probes in parallel, leading to the question of how to organize data efficiently. Here, we consider our three design targets: response time, energy consumption, and the capacity of the storage device.

Previous work [4] has studied the data layout of MEMS-based storage devices. It formulates the data-layout problem of a MEMS-based storage device in three parameters:

1) **Number of active probes:** the number of simultaneously operating probes
2) **Sector parallelism:** the number of concurrently accessible sectors from the device
3) **Sector size:** the user space storage unit

These three design targets can be enhanced rather significantly, when knowledge of the expected workload is exploited to configure the data-layout parameters. The mutual influence of the parameters on the response time, energy consumption, and capacity of MEMS-based storage devices has been studied in detail [4]. Only Pareto optimal configurations exist.

The current work investigates the data layout of MEMS-based storage devices for streaming applications exclusively. First, we carry out an exhaustive search to find the Pareto-optimal configurations of the three parameters as done for general-purpose applications. The last part exploits a unique characteristic of streaming applications, and presents a fast configuration method. We compare the configurations resulting from the fast approach to those found by the exhaustive search. The configurations are always presented as a tuple: (number of active probes, sector parallelism, sector size).

V. Experimental Methodology

Flash memories are publicly available, whereas MEMS-based storage devices are not. In this work, we adopt an empirical approach to characterize Flash memory and a trace-based simulation approach for MEMS-based storage devices.

A. MEMS Model

IBM prototyped a MEMS-based storage device of 64×64 probes [3]. Although their prototype is not publicly available for experiments, sufficient specification data are available in the literature [1]. Since MEMS-based storage devices are not yet publicly available, the second best option is to compare a real flash to a simulated MEMS-based storage device. We use trace-driven simulations. We use the DiskSim simulator [13]; a validated modular simulator for simulating various types and architectures of storage subsystems. We refine the performance and energy models of the CMU MEMS model [7] to model the IBM MEMS with better accuracy. Previous work discusses the models of timing performance, and energy consumption in detail [4], [14].

All the model parameters including, the bit dimensions and the per-probe data rate of the model are set to those from the IBM MEMS device [1]. In this work, we augment the DiskSim MEMS model with the bang-bang optimal-time seek model. This model calculates the minimum seek time, which is approximately 19% from the measured seek time of the IBM prototype [15]. This model drains the maximum current from the system, amounting to a total power dissipation of 672 mW for the X and Y electromagnetic actuators. Researchers are working on other types of actuators, such as electrostatic actuators [16], [17], to reduce the actuation energy. The relevant parameters are summarized in Table I. Since streaming workloads are predictable, data can be prefetched, and the storage device receives relatively long idle periods between requests. To prevent wasting energy in idleness, we immediately shut down the device upon request completion.

We cannot tell how exactly our MEMS model is close to reality, since we can not access a MEMS-based storage device to carry out an accuracy study. Based on our understanding

TABLE I: Settings of the simulated MEMS-based storage device

Parameter	Setting	Unit
total number of probes	64×64	probes
probe field area	100×100	μm^2
bit/track pitch	40	nm
per-probe data rate	40	Kbps
seek power	672	mW
bit access power	0.25	mW
max actuation power	120	mW
inactive power	5	mW
shutdown timeout	0	ms
number of active probes	$64 - 4096$	probes
sector parallelism	$1 - 512$	sector
sector size	$0.5 - 32$	KB

of MEMS-based storage, however, we believe that the seek model is the part that contributes with the most uncertainty to our results. This is because the seek operation involves subtle dynamics, that should be captured. In a comparison with a real device, IBM shows that the bang-bang seek model, which we adopt, is 19% off. We believe that 19% can be considered for the overall model, although the seek time is just a small portion (Figure 2b). We think that this is an overestimation, which provides a safe margin for our conclusions.

B. Storage Devices

We compare the most energy- and performance-efficient configurations of our simulated MEMS-based storage device to a SanDisk Standard CompactFlash card [18]. This CF card exhibits throughput similar to that of our simulated MEMS-based storage device. Both have throughput of approximately 10 MB/s. Further, we do not consider very-high–performance cards, such as the CF Extreme-III card, since these cards deploy multiple chips to elevate the performance, while our MEMS-based storage reference device is a basic single-sled device. In other words, we try to be as fair as possible, both in performance and cost. We cast a PDA-based setup (see next section) and characterize the timing performance and energy consumption of the adopted CF card.

In our studies, we do not consider the best-capacity configuration, because it compromises significantly on the performance. We find that smaller trade-offs can be achieved with the other two configurations, rendering them more attractive design choices. We present and discuss the configurations in Section VI-B.

C. Trace

Targeting mobile environments, we collected a representative trace on the 2 GB SanDisk Standard CompactFlash (CF) card plugged into an HP iPAQ H2215 PDA. This PDA runs an embedded version of Linux (kernel version 2.6.17). Jens Axboe's block trace utility [19] was used to log I/O events, which were forwarded to a host machine in order to minimize interference with the measurements. The CF card served as the main storage device on which the root filesystem (`rootfs`) was located. Thus, all I/O activities went from and to the CF card. The card was formatted with the `ext3` file system and a block size of 4 KB.

We turned the PDA into a multimedia device and logged several streaming scenarios. We captured a multimedia trace that includes photo taking, single and dual streaming from and to the storage device. Dual streaming represents a scenario where the user plays back a stream and downloads another one at the same time. All streaming scenarios were captured for various audio ($16 - 128$ Kbps) and video qualities ($64 - 2048$ Kbps) and with various chunk sizes ($4 - 256$ KB). We selected values from the ranges that are a power of two.

D. Assumptions

In this work, we set the bit dimensions in our MEMS model to 40 nm $\times$ 40 nm (compared to the realized 25 nm $\times$ 25 nm). Consequently, the formatted MEMS-based storage device has a capacity comparable to that of the flash card: about 2 GB. This way we maintain a fair comparison, since seeks in the MEMS-based storage device span the whole address space and thus its entire physical dimensions. Hence, we report on the worst-case seek time and seek energy. Note that the MEMS model is designed such that the change in bit dimensions affects the capacity only and not other factors, such as the data rate per probe.

To reduce its access time, a sector is striped across many probes as shown in Figure 1b. Each probe within a probe set accesses a subsector of the same sector. Striping, however, results in a capacity loss. Suppose we stripe a sector of size B bits across K probes, the subsector size b is calculated as follows:

$$b = \frac{B}{K}\,.$$

Each subsector is augmented with s synchronization bits to allow for accessibility on the subsector basis (see Figure 1b). In our model, we have $s = 3$. The number of bits per track in a storage field, n_b, and the number of tracks, n_t, are determined from the physical dimensions of the device. Let b be the size of the subsector, then the number of overhead bits, O, in one storage field is approximately:

$$O \approx \left\lfloor \frac{n_\mathrm{b} \cdot n_\mathrm{t}}{b} \right\rfloor \cdot s.$$

The right-hand side term of the equation decays nonlinearly as the subsector size increases. Here, we have $n_\mathrm{b} = \frac{100\,\mu\mathrm{m}}{40\,\mathrm{nm}} = 2500$ bits, $n_\mathrm{t} = 2500$ tracks, and $s = 3$ bits; hence the loss in capacity is only significant for $b < 100$. As an example, for $b = 12$ the capacity loss is 25%, and for $b = 100$ it is 3%. Hence, a large subsector size is highly desirable.

In the following, we search the data layout space for the best configuration from performance and energy perspectives. We compare the configurations to Flash memory and demonstrate their competitiveness. After that, we analyze the I/O subsystem in streaming applications and present a fast configuration method. We compare the resulting configurations to the optimal ones found by the exhaustive search, demonstrating the effectiveness of the fast method.

TABLE II: Best-performance (M-BP) and best-energy (M-BE) configurations of the data layout for our simulated MEMS-based storage device

Session		# probes [probe]	Sector parallelism [sector]	Sector size [KB]
32 KB	M-BP	4096	1	8
	M-BE	4096	32	4
128 KB	M-BP	4096	2	4
	M-BE	4096	32	4
combined	M-BP	4096	1	4
	M-BE	4096	32	4

VI. Comparison with Flash Memory

This section presents the configurations of our simulated MEMS-based storage device and compares them to the Flash card with respect to energy consumption and response time.

A. Comparison Methodology

From a storage-device perspective, streaming applications are characterized by three parameters: (1) the streaming direction, (2) the streaming bit rate, and (3) the streaming granularity (or prefetching unit). For example, a streaming application reads from a storage device at a bit rate of 128 Kbps, reading 16 KB of data at a time.

In streaming workloads, the streaming direction determines the operation (i.e., READ or WRITE), whereas the streaming bit rate together with the prefetching unit determines the arrival time of requests. The prefetching unit determines the address and the size of the requests. The optimal data layout of a MEMS-based storage device depends on the request address and size, and is thus influenced by the prefetching unit.

The effect of the streaming direction and streaming bit rate on the device configuration is rather straightforward; a storage device must sustain the streaming rate, and support bidirectional streaming. The streaming bit rate is achieved by increasing the number of parallel probes, bidirectional streaming requires support for alternating read and write operations. The effect of the prefetching unit is more subtle. We, therefore, focus in this work on studying the effect of scaling the prefetching unit. We consider two streaming sessions that stream from the storage device with respectively a 32 KB and a 128 KB prefetching unit. Both sessions have the same number of streaming scenarios, the same duration, and both sessions stream at constant bit rates in the range of $32 - 512$ Kbps.

B. Configurations of the Data Layout

The design space of the data layout consists of all feasible configuration of the three layout parameters presented in Section IV. Each configuration of the three parameters has three scores: a timing-performance score, a energy-consumption score, and a capacity score.

We carry out an exhaustive search to find the best configuration of the three parameters of the data layout. Table II lists the best-performance and best-energy configurations for the `multimedia` trace, one for each session, and one for the sessions combined.

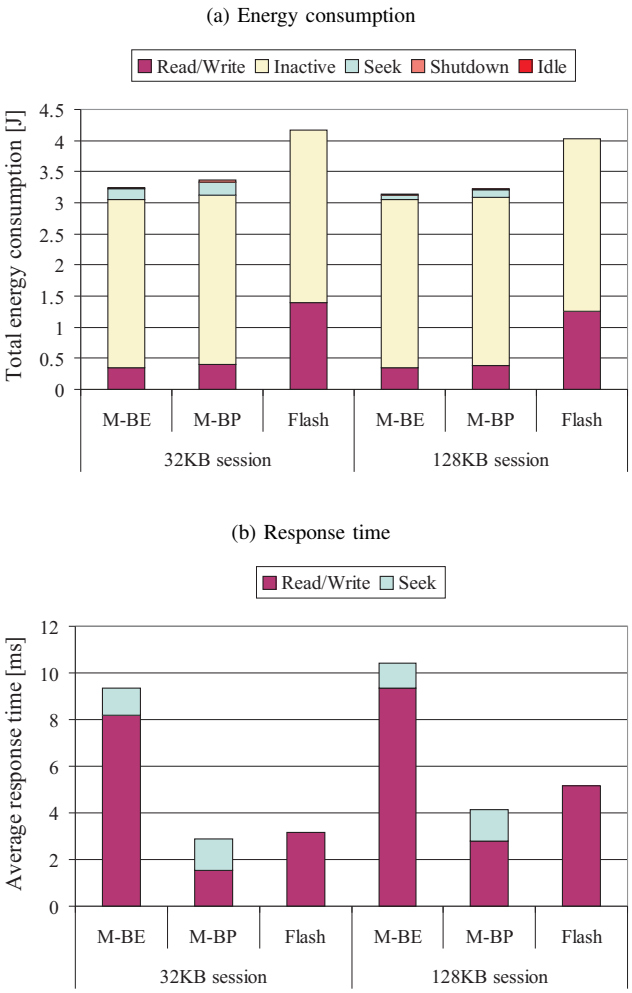


Fig. 2: Total energy consumption and average response time of the best-energy (M-BE) and best-performance (M-BP) configurations of our simulated MEMS-based storage device and the Flash card for the 32 KB and 128 KB streaming sessions

All configurations deploy 4096 probes, since increasing the number of active probes enhances the performance and energy efficiency. The table shows that the best-performance configuration varies depending on the prefetching unit. We find multiple equivalent configurations in terms of energy consumption. Table II gives the one that exhibit the best performance.

C. Comparison Results

Figure 2a shows the energy consumption and the energy breakdown of the best configurations of our MEMS-based storage device and the Flash card, three configurations for each session. Note that the seek, shutdown and idle energy of MEMS-based storage devices are negligible. The main energy component of the MEMS-based storage device as well as the Flash card is the inactive energy. This is because both technologies aggressively turn into the inactive state, where they spend most of their time. The second energy component is the read/write energy.

Figure 2a shows that our simulated MEMS-based storage device always consumes less energy than the Flash card for all scenarios. It shows that the best-energy configuration consumes approximately 22% less energy than the Flash card. Minimizing the energy consumption causes a loss in performance though. Figure 2b shows that the Flash card has between 102% and 243% shorter response time than the best-energy configuration. The best-performance configuration of our MEMS-based storage device outperforms Flash on both accounts; it has at least 8% shorter response time and consumes at least 22% less energy.

To locate the cause of the worse performance of the Flash card compared to the best-performance MEMS-based storage device, we study the distribution of the response time for every request size in the 32 KB and 128 KB session combined. Figure 3 summarizes the distributions by their average and standard deviation. It shows that the Flash card performs particularly bad for requests of size of 4 KB. A large standard deviation implies a large variation in response time. Further, because of the skewed (asymmetric) distribution of response time for the 4 KB requests, its standard deviation is even larger than the average. Other poorly performing requests sizes are 28, 32, 40, and 64 KB.

The poor performance is due to the incurred data migration that takes place in Flash to overwrite a page in a block. This migration process can take a relatively large amount of time, which explains the large response time for some request sizes. Other researchers [20] characterize the timing performance of Flash memory, and find similar variations in performance.

Figure 2b shows that the average response time with 128 KB is larger than that with the 32 KB. This is unsurprising, since the 128 KB session has larger requests than the 32 KB session. The average response time is increased by the response time of those requests of size 128 KB.

The figure shows that in either session the response time for the best-energy MEMS is significantly worse than the best-performance. This is, however, not the case with energy consumption as Figure 2a shows. The reason is that a MEMS-based storage device powers down unused probes, if they are not used to service a request. As a result, the best-energy and best-perforamnce configurations consume almost the same amount of read/write energy. On the contrary, from a response-time perspective the best-performance makes a better use of probes and avoids underutilization, since it exhibits smaller sector parallelism than the best-energy configuration as Table II shows.

VII. Configuring for Streaming Applications

In streaming environments, a single specific streaming application is continually running. Consequently, the storage device predominately services requests whose size is determined by the prefetching unit of the streaming environment. This section investigates the influence of prefetching on the performance, energy consumption, and capacity of a MEMS-based storage device as secondary storage in mobile streaming systems. Specifically, we study:

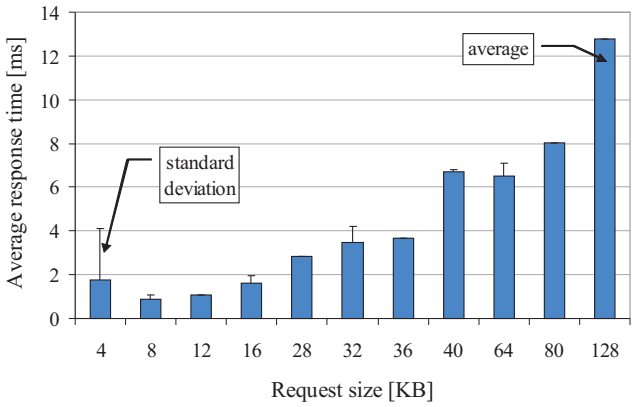


Fig. 3: The average and the standard deviation of the response time of the Flash card for each request size of the 32 KB and 128 KB session combined. Some requests have zero standard deviation (e.g., the 128 KB request size).

- the influence of upscaling the prefetching unit on the performance and energy consumption of a MEMS-based storage device (Section VII-B), and
- the direct configuration of the data layout of a MEMS-based storage device based on the prefetching unit of the streaming application (Section VII-C).

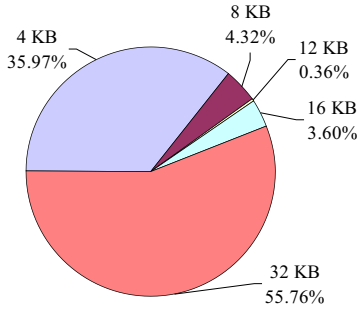
The two studies serve to determine the relation between the prefetching unit and the configuration of the data layout of a MEMS-based storage device. A good practical example, where understanding this relation can speed up the design process, is the design of streaming recording systems, such as a hand-held video camera. A MEMS-based storage device is potentially suitable for video cameras, since it promises inexpensive storage for such capacity-demanding applications. Before investigating the relation, we analyze the trace of the 32 KB session to examine the load a MEMS-based storage device services.

A. Bimodality of Request Size

Detailed analysis of the 32 KB session of the trace reveals two remarkable characteristics. The first observation is that the I/O stream is a composition of two basic streams: one from the streaming application itself and another that originates from the file system. Requests due to the file system are typically metadata writes and 4 KB in size. The second observation is that the I/O subsystem occasionally splits requests. An application request is split into two I/O requests in order to maintain fairness between applications that compete for I/O. This partitioning results in smaller request sizes than is expected from the configured prefetching unit.

A break down of the request size for the 32 KB session of the trace is given in Figure 4a. The main stream of requests has a size of 32 KB, and makes up approximately 56% of the total number of requests. Requests of size of 4 KB make up approximately 36% of the total. These requests influence the configuration of the data layout, resulting

(a) 32 KB session



(b) 128 KB session

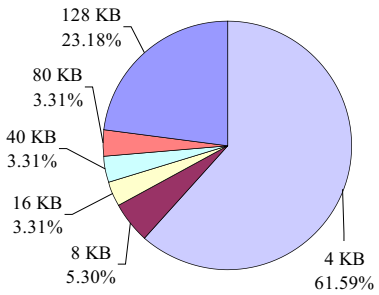


Fig. 4: Breakdown of the request size for the 32 KB and 128 KB streaming sessions. The composite stream has two main request sizes for each session, suggesting the bimodality of the distribution of the request size.

in `sector parallelism × sector size` < 32 KB for the best-performance configuration for the 32 KB session (Table II). The product `sector parallelism × sector size` is the minimum amount of data that can be accessed per request. The maximum number of simultaneously active probes is limited by power budget of the device package. For example, a SecureDigital (SD) package can afford a few hundred probes within its power budget of 0.3 W, whereas a CompactFlash (CF) card can afford a few thousand of probes within a power budget of 1.1 W.

The best-performance configuration is mainly influenced by the count of the requests, whereas the best-energy configuration is influenced by a combination of the request size and the count of the requests. As a result, since large requests count for a large part of the total energy, their influence on the configuration is dominant. This explains why the best-energy configurations in Table II have a larger product (i.e., `sector parallelism × sector size`) than that of the best-performance ones.

Summarizing, a MEMS-based storage device experiences two streams of request size due to the streaming application and the file system. Next, we discuss the influence of upscaling the request size of the streaming application.

B. Aggressive Prefetching

Streaming has a predictable nature, so that data can be prefetched to optimize for a certain resource. For example, large amounts of data are prefetched from the disk drive, so that it spins off for a long period of time to save energy. In this section, we investigate the effect of scaling the prefetching unit on the performance and energy consumption of MEMS-based storage devices. We compare the configurations for the 32 KB session and the 128 KB session. Recall that both sessions are identical in the number of streaming scenarios and the streaming bit rates; they differ only in the size of the prefetching unit.

TABLE III: Direct influence of enlarging the prefetching unit of the streaming application on the configuration of MEMS-based storage devices. A "↑" denotes an increasing trend.

		prefetching unit	⇒	sector parallelism × sector size	⇒	overhead bits
trend		↑		↓		↑
time	read/write			↓		↑
	seek					
energy	read/write			↓		↑
	seek	↓				
	inactive	↑		↑		↓
	shutdown	↓				
	idle					
capacity						↓

Increasing the prefetching unit results in fewer accesses to the storage device. Consequently, both the seek energy (to seek from the center to the position of the requested data) and the shutdown energy (to bring the media sled back to the center) decrease, although they are insignificant. Since the device is accessed fewer times, it spends more time in inactivity. The first column in Table III indicates the effects on the response time and energy consumption.

As for the best-performance configuration, increasing the prefetching unit results in a smaller minimum request size, i.e., `sector parallelism × sector size`. Table II shows that the sector size is 8 KB when the prefetching unit is 32 KB, whereas it is 4 KB when the prefetching unit is 128 KB. Although one would expect the sector size to increase when the prefetching unit increases, this is not the case. The reason is that when the prefetching unit increases, requests due to the streaming application become larger and fewer. As a result, the percentage of requests due to the file system increases, which results in a larger influence of filesystem requests in determining the best performing configuration. Figure 4a and Figure 4b show that the percentage of requests of size of 4 KB increases from 36% for the 32 KB session to 62% for the 128 KB session, while their absolute count are equivalent (100 and 93 requests, respectively).

The second column in Table III shows the influence of decreasing the minimum request size `sector parallelism × sector size` on the response time and energy consumption of MEMS-based storage devices. Decreasing the minimum request size decreases the read/write time, and thus reduces the read/write energy. The reason is that when the sector size (or the sector parallelism) decreases, the utilization of probes increases, since it is more likely that probes access useful data.

Reducing the sector size (and thus the subsector size), however, results in more overhead bits per sector as explained in Section V-D. As a result, the effective capacity of the device decreases as shown in the third column in Table III. Further, more time and energy are invested to read these bits. The overhead can be significant; the capacity of the best-performance configuration is 1.99 GB compared to 2.61 GB of the best-capacity configuration.

Like the best-performance configuration, the best-energy configuration does not benefit from a large prefetching unit. Figure 2a shows that the best-energy configuration of our simulated MEMS-based storage device consumes the same amount of energy for the two prefetching units. This is because of the negligible seek and shutdown energy, so that savings on these components are unpronounced.

In summary, MEMS-based storage devices are negatively influenced when the prefetching unit increases. When the prefetching unit is large, a significant capacity loss is incurred, if the data layout is configured with small sectors. On the other hand, if the layout is configured with sectors equally sized as the prefetching unit, the response time of small requests increases significantly. Furthermore, a large prefetching unit gains the device a negligible amount of energy.

C. Exploiting the Bimodality

We investigate the configuration of MEMS-based storage devices by exploiting the bimodality in the request size in streaming environments. We show that such a configuration methodology, based on the bimodality property, results in configurations that are close to the Pareto-optimal configurations discussed in Section VI-B, which were found by an exhaustive exploration of the design space.

Recall from the previous section that increasing the sector size directly reduces the overhead per sector, and enhances the performance and energy efficiency. On this basis, we investigate setting the sector size to one of the two main sizes of the bimodal distribution of the request size. We fix the sector parallelism to one sector all the time. We simulate for the 32 KB session and the 128 KB session.

Figure 5 presents the improvement/degradation factor when configuring with one of the main request sizes for both sessions. The values are normalized to the best configuration for each respective design target for that session. That is, the energy figures for the two configurations of the 32 KB session, for instance, are normalized to the best-energy configuration for the 32 KB session listed in Table II. Unlike energy consumption and response time, the lower the normalized value of the capacity, the smaller the capacity and thus less favorable. The best overall configuration is the one that keeps the score on all targets close to one.

Observe that the difference in energy consumption between the two configurations is negligible for both sessions. Figure 5 shows that the response time worsens drastically when formatting with a large sector size as for the 128 KB session; about 2.5 times longer response time than the best-performance configuration. The reason is that the increase in the sector

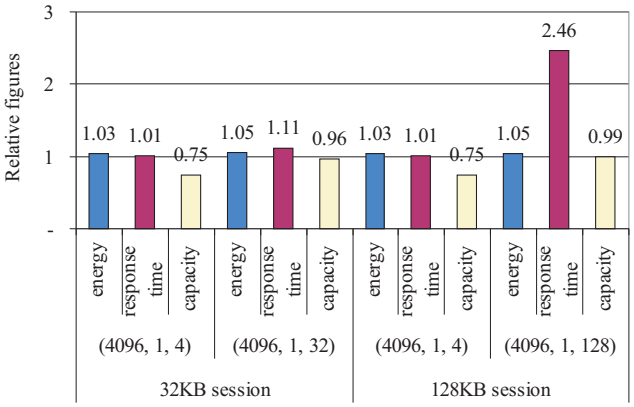


Fig. 5: Possible configurations of a MEMS-based storage device based on the bimodal distribution of the request size in streaming environments. The sector size is set to one of the two main request sizes. The best overall configuration is the one that keeps the score on all targets close to one, such as (4096, 1, 32).

size, increases the underutilization of probes when servicing small requests. On the contrary, formatting with a large sector size reduces the overhead per sector, and thus increases the effective capacity.

Comparing the scores on the three targets of the 32 KB session to the 128 KB, one reveals that aggressive prefetching on the application level leads to a loss in either the capacity (case: 4096 probes, 1 sector, 4 KB) or the performance (case: 4096 probes, 1 sector, 128 KB). Decreasing the prefetching unit enables the designers to reach smaller trade-offs between the capacity and the performance (case: 4096 probes, 1 sector, 32 KB) for the 32 KB session; 5% more energy, 11% longer response time, and 4% less capacity compared to the best configurations, respectively.

The bimodality results in configurations that are competitive to the best ones found by the exhaustive search, provided that the prefetching unit is regarded. The configuration of (4096 probes, 1 sector, 32 KB) is the resultant one in our example.

In summary, designers of MEMS-based storage devices can exploit the bimodality in streaming environments to find configurations *with small trade-offs quickly*. Further, decreasing the size of the prefetching unit of streaming applications enables the designer to find configurations that have smaller trade-offs between the design targets than what is possible with large prefetching units.

D. A Configuration Method

The study of the configuration of the data layout presented thus far in this section concludes with the following configuration method of the data layout of MEMS-based storage devices in streaming environments:

1) Model the workload as a composition of two streams with request sizes A and B. Request size A corresponds

to requests due to the file system. Request size B corresponds to the size of the prefetching unit.

2) Emulate the system with all possible configurations. Set the data-layout parameters as follows:
   `number of probes = maximum allowed`,
   and set:
   `sector size × sector parallelism =` A,
   or
   `sector size × sector parallelism =` B.

3) Select the most beneficial configuration from the above two based on the design target(s).
4) Scale down the prefetching unit to evaluate other possible trade-offs as done in steps 2 and 3.
5) Finally, select the configuration that scores best on the design target(s).

Generalization of this method to general purpose applications is not possible, because of the large variation of the request size in these environments.

VIII. Conclusions

This work targets employing MEMS-based storage devices in mobile streaming systems. We show that formatting the data layout based on the expected workload makes MEMS-based storage devices at least 22% more energy efficient than Flash memory, while exhibiting a comparable performance.

Our study reveals that I/O requests follow a bimodal distribution in their size. We make the case for exploiting the bimodality property and show that the designer can effectively format the data layout. Doing so, the configuration process is fast, since the designer avoids an exhaustive space exploration.

Our results support reducing the amount of prefetching of streaming data to allow for near-optimal configurations. Such configurations have small trade-offs between the response time, energy consumption, and capacity of MEMS-based storage devices. Further, these configurations are comparable to the best configurations found by an exhaustive search.

Acknowledgments

This research is supported by the Technology Foundation STW, applied science division of NWO, the technology programme of the Ministry of Economic Affairs under project number TES.06369, and by the European Research Council under a FP-6 grant for the project ProTeM.

References

[1] M. A. Lantz, H. E. Rothuizen, U. Drechsler, W. Häberle, and M. Despont, "A Vibration Resistant Nanopositioner for Mobile Parallel-Probe Storage Applications," *Journal of Microelectromechanical Systems*, vol. 16, no. 1, pp. 130–139, February 2007.

[2] Nanochip, "Nanochip develops MEMS-based storage." http://nanochipinc.com/tech.htm, 2008, accessed in November 2007.

[3] A. Pantazi, A. Sebastian, T. A. Antonakopoulos, P. Bächtold, A. R. Bonaccio, J. Bonan, G. Cherubini, M. Despont, R. A. DiPietro, U. Drechsler, U. Dürig, B. Gotsmann, W. Häberle, C. Hagleitner, J. L. Hedrick, D. Jubin, A. Knoll, M. A. Lantz, J. Pentarakis, H. Pozidis, R. C. Pratt, H. Rothuizen, R. Stutz, M. Varsamou, D. Wiesmann, and E. Eleftheriou, "Probe-based ultrahigh-density storage technology," *IBM Journal of Research and Development*, vol. 52, no. 4/5, pp. 493–511, July/September 2008.

[4] M. G. Khatib, E. L. Miller, and P. H. Hartel, "Workload-based Configuration of MEMS-Based Storage Devices for Mobile Systems," in *Proceedings of the 8th ACM & IEEE International Conference on Embedded Software (EMSOFT '08), Atlanta, USA*. USA: ACM, October 2008, pp. 41–50.

[5] L. R. Carley, J. A. Bain, G. K. Fedder, D. W. Greve, D. F. Guillou, M. S. C. Lu, T. Mukherjee, S. Santhanam, L. Abelmann, and S. Min, "Single-chip computers with microelectromechanical systems-based magnetic memory (invited)," *Journal of Applied Physics*, vol. 87, no. 9 III, pp. 6680–6685, 2000. [Online]. Available: www.scopus.com

[6] L. Abelmann, T. Bolhuis, A. M. Hoexum, G. J. M. Krijnen, and J. C. Lodder, "Large capacity probe recording using storage robots," *IEE Proceedings: Science, Measurement and Technology*, vol. 150, no. 5, pp. 218–221, 2003. [Online]. Available: www.scopus.com

[7] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle, "Modeling and performance of MEMS-based storage devices," in *Proceedings of ACM SIGMETRICS 2000*, Santa Clara, California, 17-21 June, 2000, pp. 56–65. [Online]. Available: http://www.pdl.cmu.edu/PDL-FTP/Storage/sigmetrics2000{_}abs.html

[8] F. Wang, B. Hong, S. A. Brandt, and D. D. E. Long, "Using MEMS-based storage to boost disk performance." in *MSST'05: 22nd IEEE Conference on Mass Storage Systems and Technologies*, April 2005, pp. 202–209.

[9] R. Rangaswami, Z. Dimitrijevic, E. Chang, and K. E. Schauser, "MEMS-based disk buffer for streaming media servers," *Proceedings of 19th International Conference on Data Engineering*, pp. 619–630, March 2003.

[10] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger, "Designing Computer Systems with MEMS-based Storage," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2000, pp. 1–12. [Online]. Available: http://citeseer.ist.psu.edu/schlosser00designing.html

[11] M. Uysal, A. Merchant, and G. A. Alvarez, "Using MEMS-Based Storage in Disk Arrays," in *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, March 31 – April 2 2003, pp. 89–101.

[12] E. Eleftheriou, T. Antonakopoulos, G. K. Binnig, G. Cherubini, M. Despont, A. Dholakia, U. Dürig, M. A. Lantz, H. Pozidis, H. E. Rothuizen, and P. Vettiger, "Millipede—A MEMS-Based Scanning-Probe Data-Storage System," *IEEE Transactions on Magnetics*, vol. 39, no. 2, pp. 938–945, Mar. 2003. [Online]. Available: http://www.zurich.ibm.com/pdf/sys/storage/ELE{_}IEEE{_}Trans{%}20Mag2003.pdf

[13] H. S. Bucy, G. R. Ganger, and Contributors, "The DiskSim simulation environment version 3.0," School of Computer Science, Carnegie Mellon University, Reference Manual, January 2003.

[14] M. G. Khatib and P. H. Hartel, "Power Management of MEMS-Based Storage Devices for Mobile Systems," in *Proceedings of the 8th ACM & IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '08), Atlanta, USA*. Danvers: ACM, October 2008, pp. 245–254.

[15] A. Sebastian, A. Pantazi, G. Cherubini, M. Lantz, H. Rothuizen, H. Pozidis, and E. Eleftheriou, "Towards faster data access: Seek operations in MEMS-based storage devices," *Control Applications, 2006. CCA '06. IEEE International Conference on*, pp. 283–288, October 2006.

[16] J. B. C. Engelen, H. E. Rothuizen, U. Drechsler, R. Stutz, M. Despont, and M. A. Lantz, "Mass-balanced through-wafer electrostatic x/y-scanner for parallel probe data storage," in *34th International Conference on Micro & Nano Engineering, Athens, Greece*. Greece: Ergo Publications, 2008, p. 92.

[17] I. Chung, J. Jeon, and K. Park, "Design of an Electrostatic 2-axis MEMS Stage with Large Area Platform for PSD," in *Science and Technology, 2005. KORUS 2005. Proceedings. The 9th Russian-Korean International Symposium on*, June-2 July 2005, pp. 466–470.

[18] "SanDisk Standard CompactFlash card," http://www.sandisk.com/OEM/ProductCatalog(1337)-CompactFlash_Memory_Card.aspx, accessed in April 2007.

[19] Kernel Tracing, "Kernel Trace Systems," http://elinux.org/Kernel_Trace_Systems, 2009, accessed in November 2007.

[20] A. Birrell, M. Isard, C. Thacker, and T. Wobber, "A design for high-performance flash disks," *SIGOPS Operating Systems Review*, vol. 41, no. 2, pp. 88–93, April 2007.